Generating Certificates

Revised: March 27, 2006, OL-8880-01

Overview

This chapter provides a general overview of the steps involved in generating RSA keys and certificates without reference to specific tools. Following the overview, the sections Generating Certificates with OpenSSL, page 2-2 and Certificate Generation with Windows CA, page 2-6 provide examples based on OpenSSL and Windows Certificate Authority.

The actual mechanics of certificate generation are highly dependent on the tools used as well as the local security policies in effect. Some tools and policies might condense the three steps shown below into fewer (possibly one) steps or expand them into more steps. The degree of automation and direct user involvement also varies greatly and can range from a simple web form-based model with automatic certificate distribution to a more complicated procedure with multiple user interactions. Some CAs are set up to support online operations including certificate production while others might operate strictly offline and require more manual involvement.

RSA Key Generation

RSA keys have certain mathematical and cryptographic properties that require special software tools for the generation. Some tools will ask you to type on the keyboard during generation to create a source of randomness. This is because RSA keys are based on large random numbers.

RSA key pairs have two essential parameters that must be specified during creation. The first parameter is the key type which is always RSA. The second parameter is the key length in bits which can vary from 512 to 4096 bits (or even more). The key length is usually specified as part of the customers' security policy and it is difficult to give a generally applicable recommendation for it.

Certificate Request Creation

A Certificate Request (CR) is information packaged with the public key that specifies the type and general content of the desired certificate. It is usually packaged in a format based on PKCS#10 (one of the PKCS standards documented by RFC 2986) or Certificate Request Message Format (CRMF), an emerging standard from the IETF. The format of the CR is usually not important as long as the tools used to create and process it are compatible.

The CR usually contains the following:

- An RSA key-pair
- Subject name (possibly in DN format)
- Desired lifetime of the certificate
- Name or identification of the issuing (signing) CA
- Certificate extensions

PEAP and EAP-TLS require server certificates to include an extendedKeyUsage extension of *TLS Server Authentication* and client certificates to include an extendedKeyUsage extension of *TLS Client Authentication*. The method used to specify extensions and their values depends on the tool. The extendedKeyUsage extension contains one or more Object Identifier (OID) values which are specified as strings of dot-separated decimals. The appropriate values are shown below:

Table 2-1 Required Values of extendedKeyUsage

Server or Client	OID Value	Meaning
Server	1.3.6.1.5.5.7.3.1	TLS Server Authentication
Client	1.3.6.1.5.5.7.3.2	TLS Client Authentication

Certificate Generation

The CR is submitted to the CA to generate the actual certificate. This might happen immediately, upon request (such as using a web form) or there might be delays if the CA is operated offline or requires manual management approval to issue certificates.

Generating Certificates with OpenSSL

This section provides example of creating certificates with OpenSSL. The OpenSSL open source project includes a command line tool, **openssl**, used to create keys and certificates. OpenSSL has many other useful capabilities. For more information about the OpenSSL open source project, check their website:

http://www.openssl.org

The examples create an extremely simple certificate hierarchy consisting of two levels and three certificates. First a self-signed root certificate is created and then used to sign a server certificate and a client certificate. Most realistic certificate hierarchies contain one or more levels of intermediate CA certificates.

The following steps assume a Linux or BSD-style shell is active, but the commands for most other command line environments are similar.

The openssl.cnf Configuration File

The **openssl** command tool uses a configuration file usually named **openssl.cnf** for various parameters and other information related to creating certificate requests.

There are two ways to make the **openssl.cnf** file available to the command tool:

- Using the OPENSSL_CONF environment variable

export OPENSSL_CONF /opts/open/openssl.cnf

- Specifying the *–config* option on the command line

  openssl <*additional parameters*> config ./openssl.cnf

The CA and CA_Default sections of the **openssl.cnf** file are particularly important because they specify information related to the configuration of a CA. Figure 2-1 shows an example of an **openssl.cnf** file, from the OpenSSL distribution.

Figure 2-1 Example openssl.cnf File

```
####################################################################
[ ca ]
default_ca      = CA_default                # The default ca section

####################################################################
[ CA_default ]

dir             = ./ca                      # Where everything is kept
certs           = $dir/cert                 # Where the issued certs are kept
crl_dir         = $dir/crl                  # Where the issued crl are kept
database        = $dir/index.txt            # database index file.
new_certs_dir   = $dir/newcerts             # default place for new certs.

certificate     = $dir/root-cert.pem        # The CA certificate
serial          = $dir/serial               # The current serial number
crl             = $dir/crl.pem              # The current CRL
private_key     = $dir/private/root-key.pem  # The private key
RANDFILE        = $dir/private/.rand        # private random number file

x509_extensions = usr_cert                  # The extentions to add to the cert

# Comment out the following two lines for the "traditional"(and highly broken) format.
name_opt        = ca_default                # Subject Name options
cert_opt        = ca_default                # Certificate field options

# Extension copying option: use with caution.
# copy_extensions = copy

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crl_extensions  = crl_ext

default_days      = 365                     # how long to certify for
default_crl_day  s= 30                      # how long before next CRL
default_md        = md5                     # which md to use.
preserve          = no                      # keep passed DN ordering

# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy            = policy_match
```

Required Certificate Extensions

PEAP and EAP-TLS require server certificates to include an extendedKeyUsage extension of *TLS Server Authentication* and client certificates to include an extendedKeyUsage extension of *TLS Client Authentication*. These extensions can be placed in a configuration file referenced on the **openssl** command line.

The following is an example of the required **certs-exts.cnf** extensions file:

```
[ server_exts ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.1
[ client_exts ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.2
```

Creating Test Certificates and Keys

Use the **openssl** command line tool to create certificates and keys for testing PEAP. The following sections provide examples of how to create a simple certificate hierarchy that consists of a single CA certificate, a single server certificate, and a single client certificate. Additional certificates and keys can be produced as needed for testing purposes.

Note Long commands are shown on multiple lines, and some of the commands will prompt you for additional input.

Creating a CA Directory

To create a CA directory, enter the following commands as a root user:

mkdir ca

cd ca

mkdir certs private reqs

echo '01' > serial

touch index.txt

chmod 0700 private

cd ..

Creating a Self-signed CA Root Certificate and RSA Key

Use the following command sequence to create a self-signed CA root certificate and RSA key.

openssl req -x509 -newkey rsa:1024 –keyout ./ca/private/root-key.pem -keyform PEM -out ./ca/certs/root-cert.pem -outform PEM -config ./openssl.cnf

Use the following command to display the certificate:

openssl x509 –in ./ca/certs/root-cert.pem –text

Converting a CA Certificate to PKCS#12

Use the following command sequence to convert a CA certificate to PKCS#12 format. This process is useful for importing a CA certificate to a Windows PC for testing purposes.

cat ./ca/certs/root-cert.pem ./ca/private/root-key.pem > ./ca/private/root-all.pem

openssl pkcs12 -export -in ./ca/private/root-all.pem -out ./ca/certs/root-cert.p12

Creating a Server Certificate Request and RSA Key

Use the following command sequence to create a server certificate request and RSA key.

openssl req -newkey rsa:1024 -keyout ./ca/private/server-key.pem -keyform PEM -out ./ca/reqs/server-req.pem -outform PEM -config ./openssl.cnf

Creating a Server Certificate from the Request

Use the following command sequence to create a server certificate from the request and reference the certificate extensions file and required server certificate extension.

openssl x509 -req -days 365 -in ./ca/reqs/server-req.pem -CA ./ca/certs/root-cert.pem -CAkey ./ca/private/root-key.pem -CAserial ./ca/serial -extfile ./ca/cert-exts.cnf -extensions server_exts -out ./ca/certs/server-cert.pem

Use the following command to display the server certificate:

openssl x509 –in ./ca/certs/server-cert.pem –text

Creating a Client Certificate Request

Use the following command sequence to create a client certificate request.

openssl req -days 365 -newkey rsa:1024 -keyout ./ca/private/client-key.pem -keyform PEM -out ./ca/reqs/client-req.pem -outform PEM -config ./openssl.cnf

Creating a Client Certificate from the Request

Use the following command sequence to create a client certificate from the request and reference the certificate extensions file and required client certificate extension.

openssl x509 -req -days 365 -in ./ca/reqs/client-req.pem -CA ./ca/certs/root-cert.pem -CAkey ./ca/private/root-key.pem -CAserial ./ca/serial -extfile ./ca/cert-exts.cnf -extensions client_exts -out ./ca/certs/client-cert.pem

Use the following command to display the server certificate:

openssl x509 –in ./ca/certs/client-cert.pem –text

Converting a Client Certificate and Private Key to PKCS#12

Use the following command sequence to convert a client certificate and private key to PKCS#12. This process is useful for importing a client certificate to a Windows PC for testing.

cat ./ca/certs/client-cert.pem ./ca/private/client-key.pem > ./ca/private/client-all.pem

openssl pkcs12 –export –in client-all.pem –out client-all.p12

Certificate Generation with Windows CA

This section provides examples of creating certificates using the Windows Certificate Authority (Windows CA). The Windows CA provides a web-based interface for requesting and retrieving certificates. The web forms permit you to create a new key pair or use an existing key, specify the desired certificate fields and attributes, and to submit the request to the CA for processing.

Note The Windows CA component is only available on Windows Server OS, not on client OS (such as Windows 2000 Pro or Windows XP). To generate certificates you will need a Windows Server set up and the Windows CA configured.

Usually an administrator will be required to manually review and grant or deny the request before the certificate can be accessed. (Windows CA can also be configured to automatically grant requests without administrator intervention.) The Certification Authority snap-in of the Microsoft Management Console (MMC) is used to review certificate requests and take the appropriate action. It can also be used for other purposes such as certificate revocation, renewal, etc.

After a certificate has been issued by the Windows CA it must be exported to a file so that it can be transported to the machine where it will be used. Although Windows can export certificates in DER or PEM format, if the corresponding private key is required (as it is for server and client certificates) then the certificate and private key will be bundled into a PKCS#12-formatted file. Since the required format for our purposes is PEM, the PKCS#12 content must be reformatted appropriately.

The following examples show an extremely simple certificate hierarchy consisting of two levels and three certificates. Most realistic certificate hierarchies will contain one or more levels of intermediate CA certificates. Since the root-level certificate is created when the Windows CA product is installed and configured, those steps are not shown here. The examples assume that the Windows CA has been configured for standalone operation, but the steps are essentially the same for other configurations.

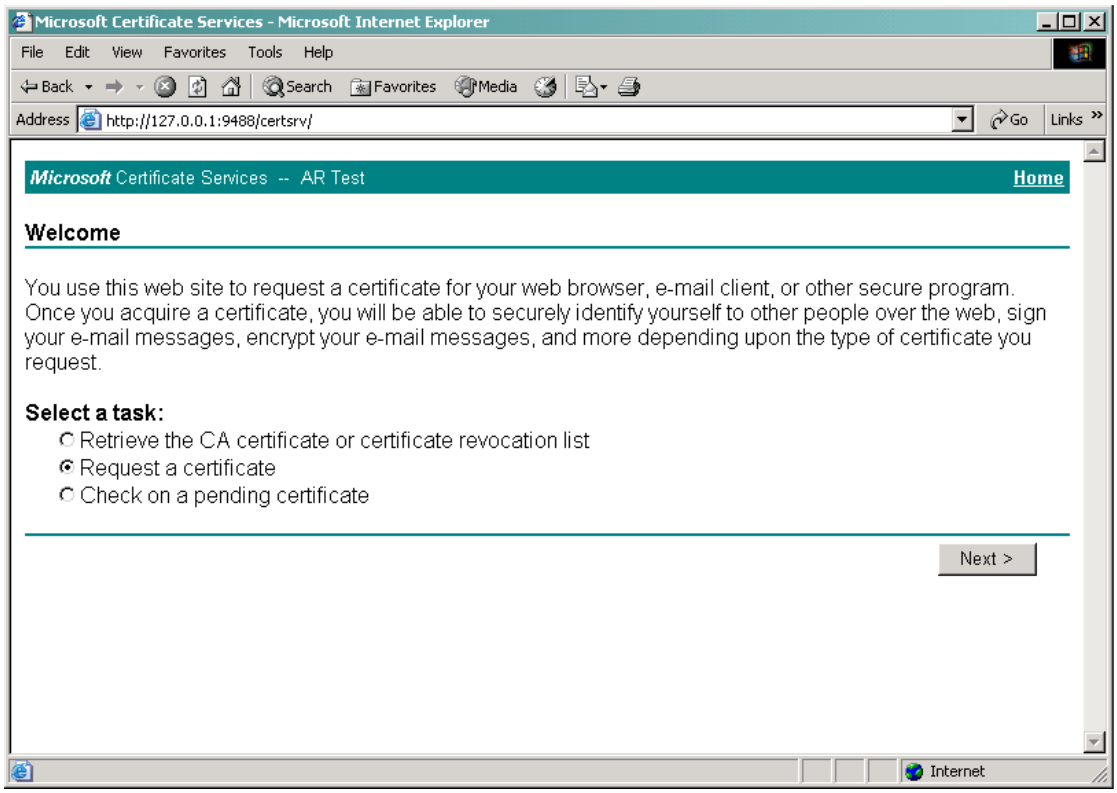
The following examples assume that the Windows Certificate Authority product has been installed and configured. Since the exact installation steps vary depending on the version of Windows Certificate Authority and its configuration, those steps are not shown here. Refer to the appropriate Microsoft documentation for information about how to install Windows Certificate Authority.

Generating a Server Certificate

This section describes how to generate a server certificate.

Step 1 Use your browser to access the Windows Certificate Services web form using a URL like the following: **http://server-name/certsrv**. In the example below, the server name is w2ks.

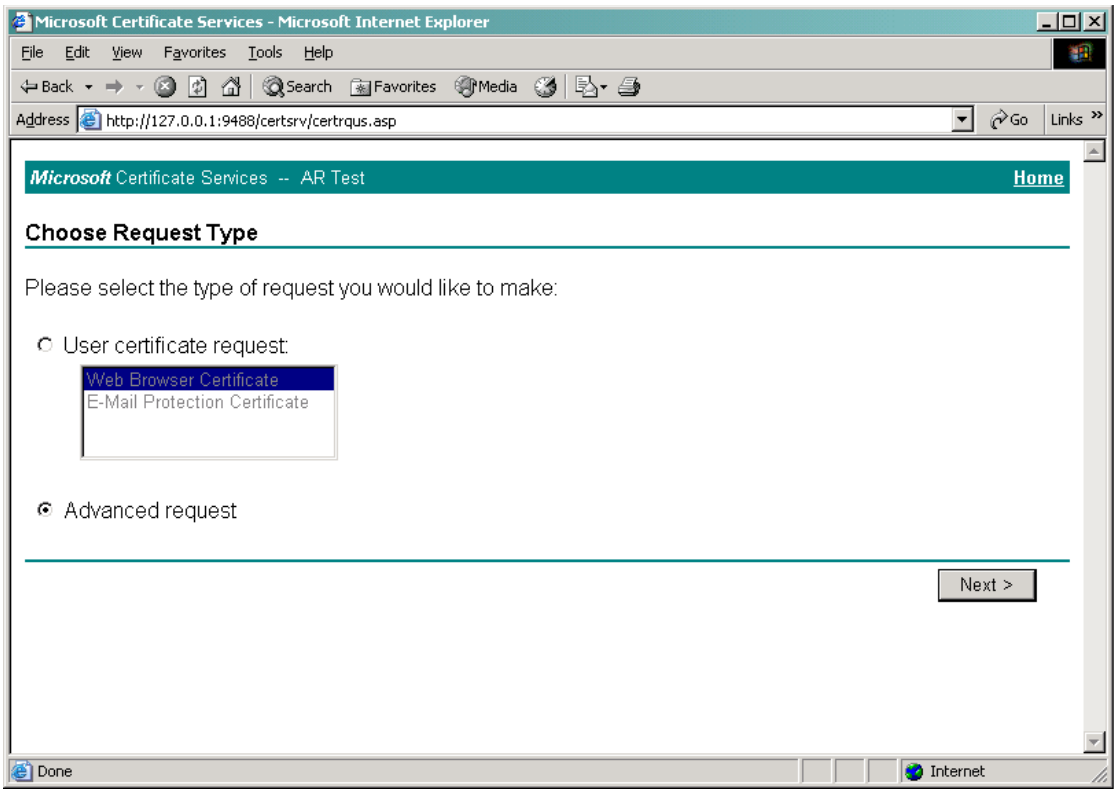
Figure 2-2 Windows Certificate Services Form



Step 2 Select **Request a Certificate** and click **Next**.

The next window enables you to select the type of certificate request.

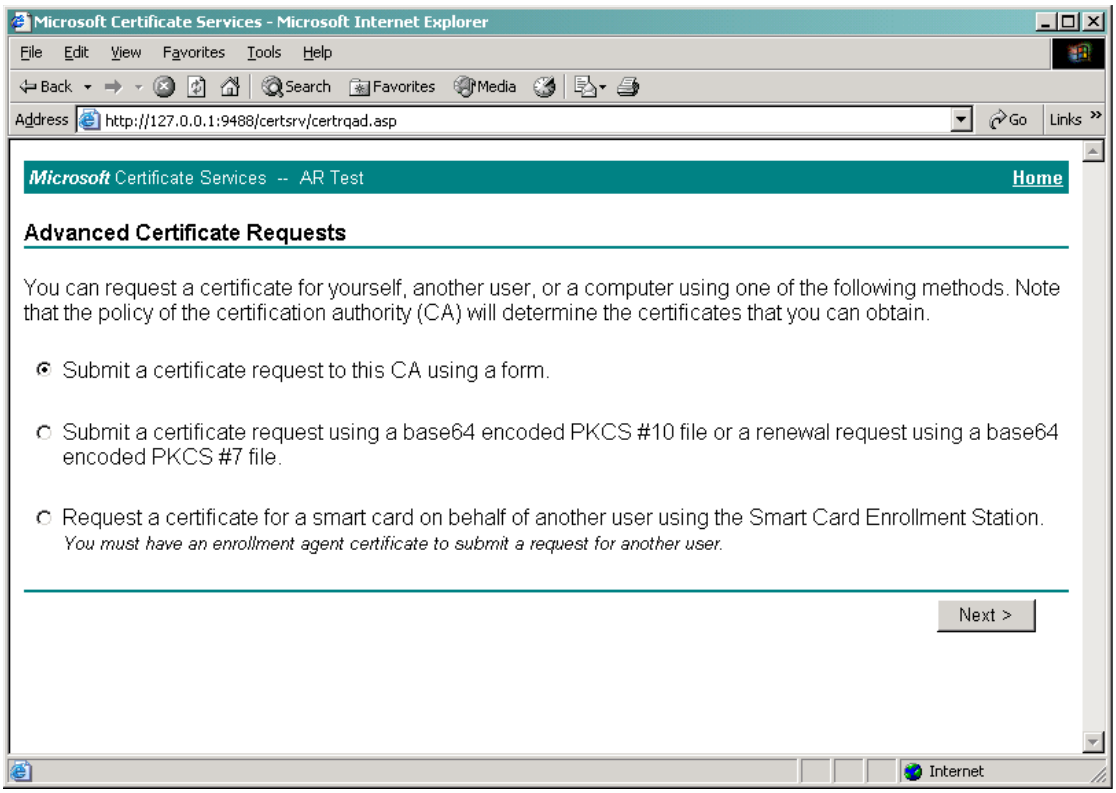
Figure 2-3 Selecting Certificate Request Type



Step 3 Select **Advanced request** and click **Next**.

The next window enables you to select the method used to request the certificate.

Figure 2-4 Advanced Certificate Requests



Step 4 Select **Submit a certificate request to the CA using a form**, then click **Next**.

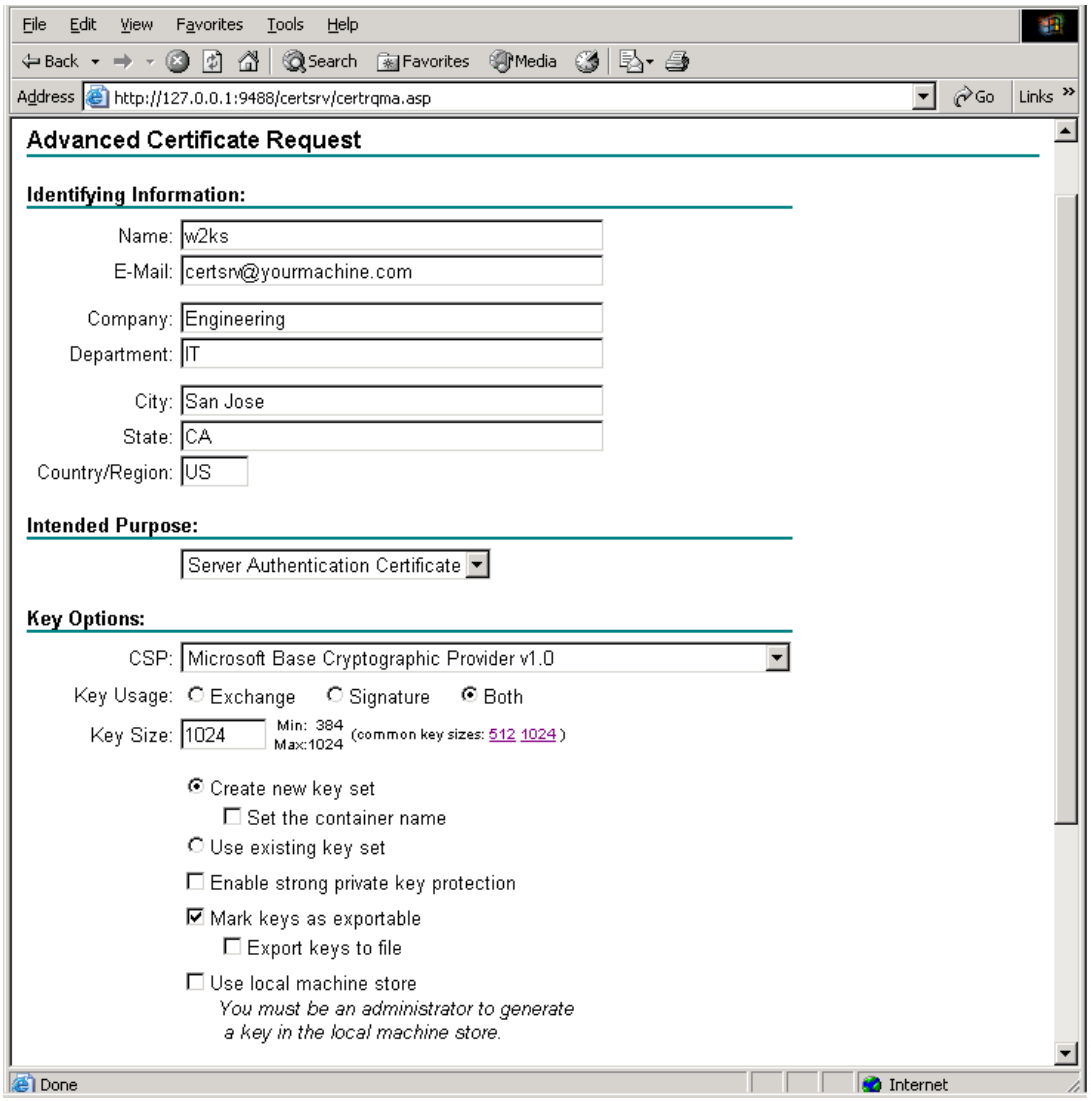
The Advanced Certificate Request form, Figure 2-5, allows you to specify some of the certificate's information content. You need only specify a few items; use the default values for the others.

In the *Identifying Information* section, **Name** is usually the name of the server. Use the default values of the other fields.

In the *Intended Purpose* section, select **Server Authentication Certificate**.

In the *Key Options* section, select **Mark keys as exportable**.

Figure 2-5 Advanced Certificate Request Form

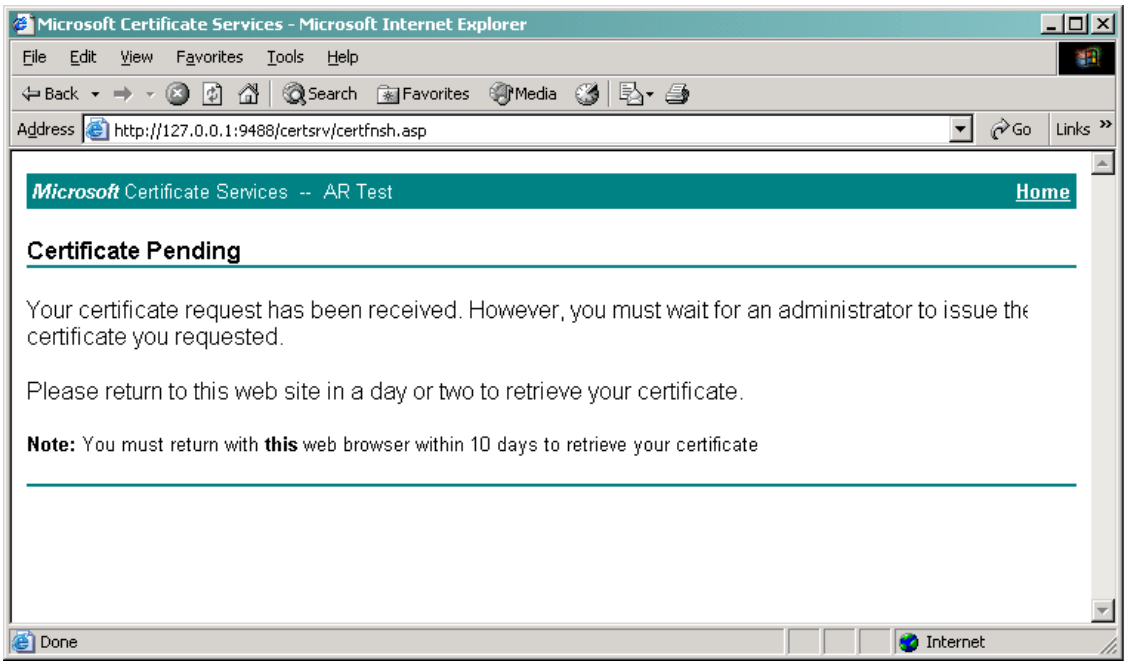


Step 5 After you provide the information required for the form, click **Submit** to submit the request.

Note Depending on your configuration, you might be asked to confirm your request.

The next window acknowledges receipt of the request and advises you to check back later to retrieve the certificate.

Figure 2-6 Certificate Pending



Step 6 Click **Home** (near upper right corner of form) to return to the **Certificate Services** home page.

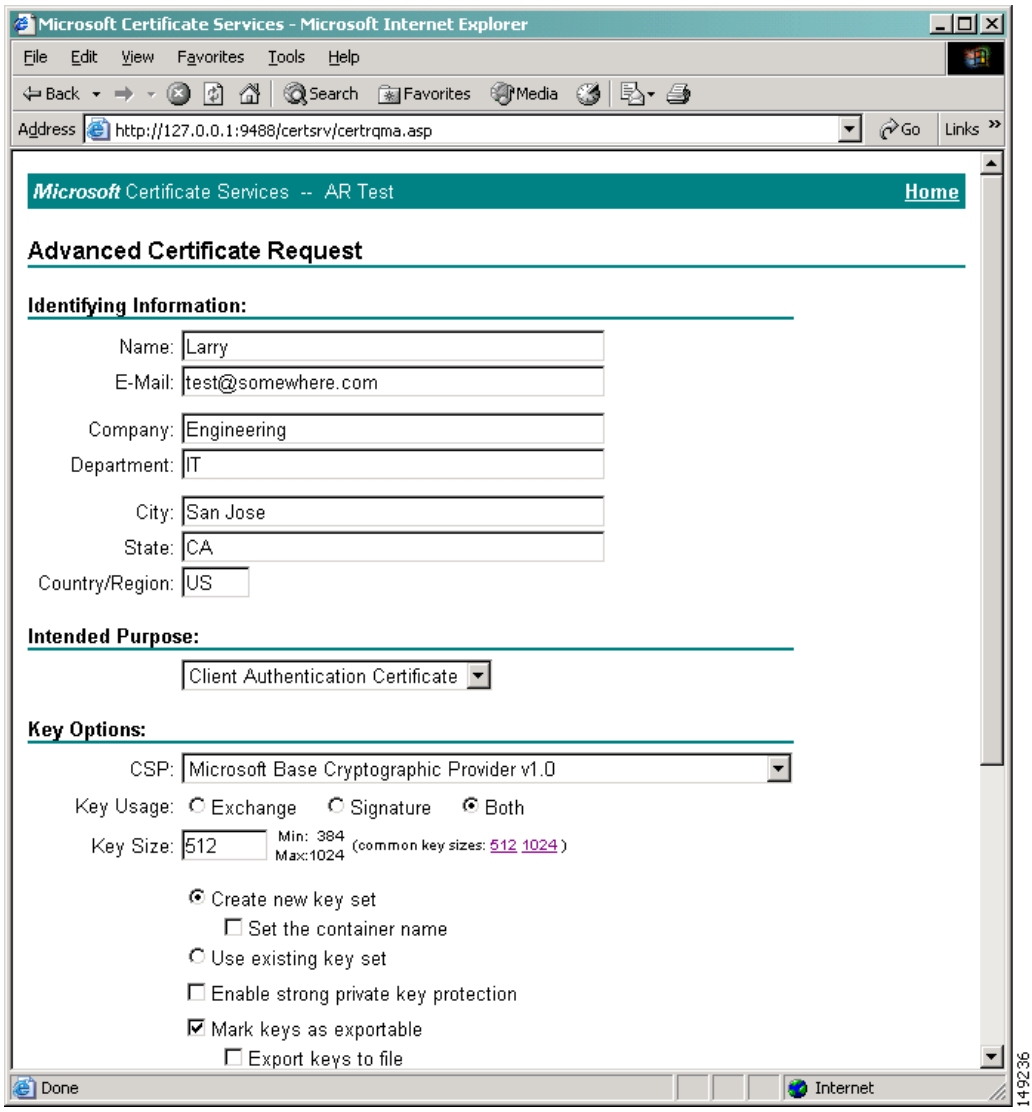
Generating a Client Certificate

The procedure to generate a client certificate is very similar to the procedure to generate a server certificate. The only significant differences are the value of the Name field and the Intended Purpose on the Advanced Certificate Request page.

Because this is a client certificate, the Name field should contain the user ID if the certificate is for an individual or the machine name if the certificate is for a computer. The value of the Intended Purpose field must be set to Client Authentication Certificate.

Figure 2-7 shows an example of the Advanced Certificate Request form for requesting a client certificate.

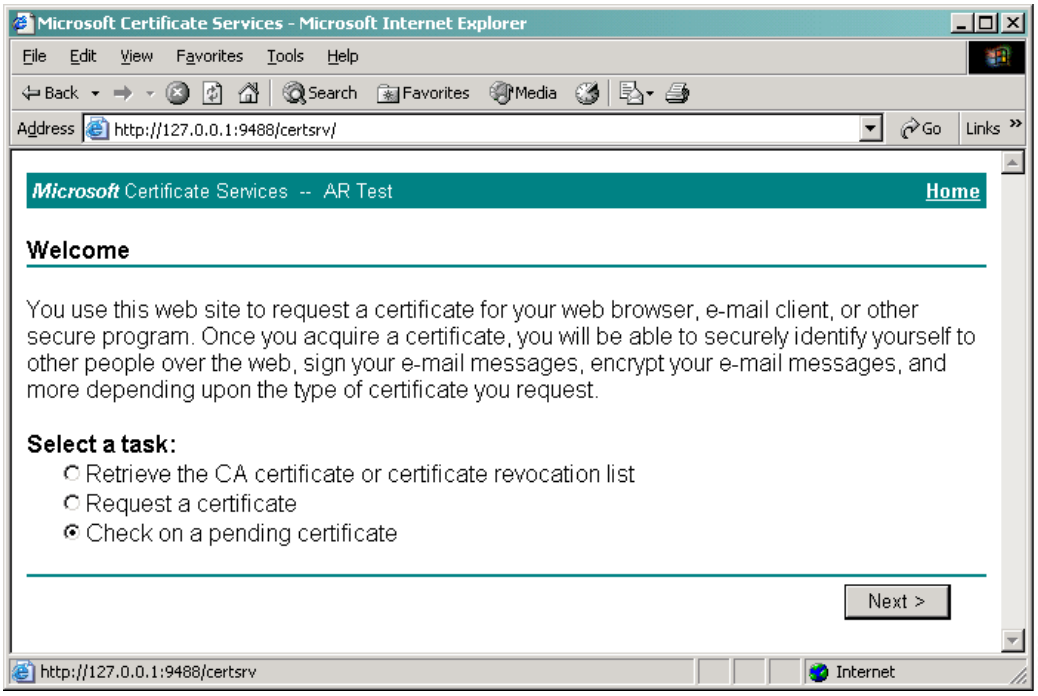
Figure 2-7 Example of Client Certificate Request Form



Certificate Retrieval

From the Certificate Services home page, select **Check on a pending certificate**.

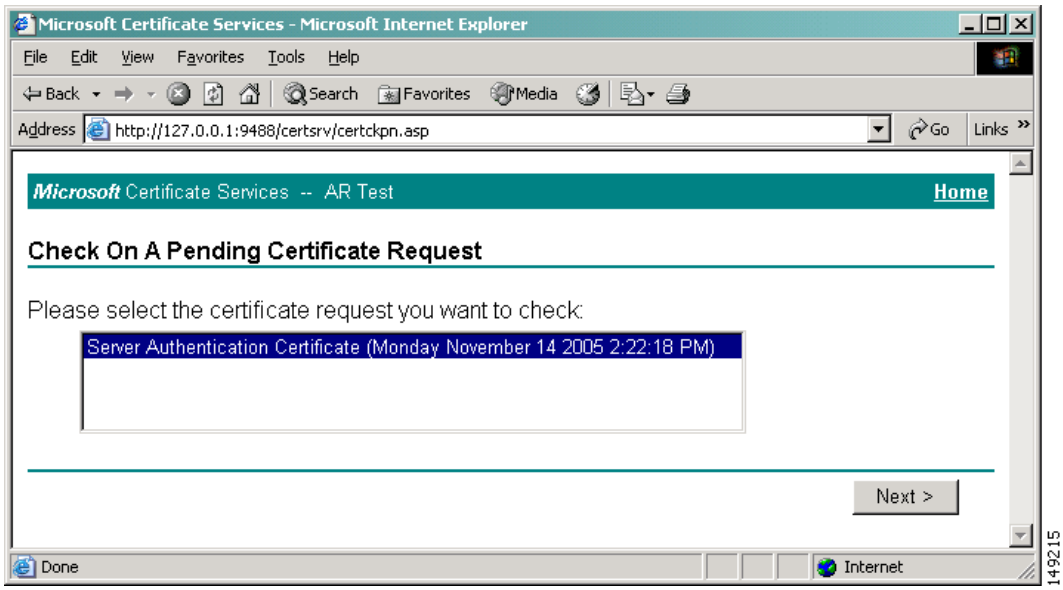
Figure 2-8 Example of Check Pending Certificate Request



Step 7 Click **Next** to proceed.

Figure 2-9 shows an example of the pending certificates requests.

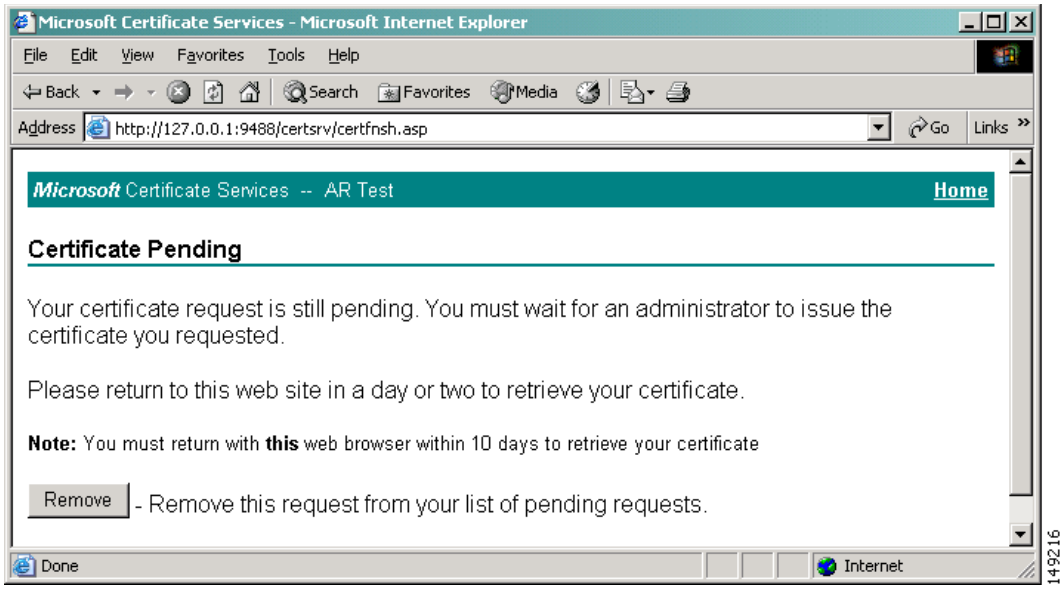
Figure 2-9 Check Pending Certificate Requests



Step 8 Select the appropriate request from the list and click **Next**.

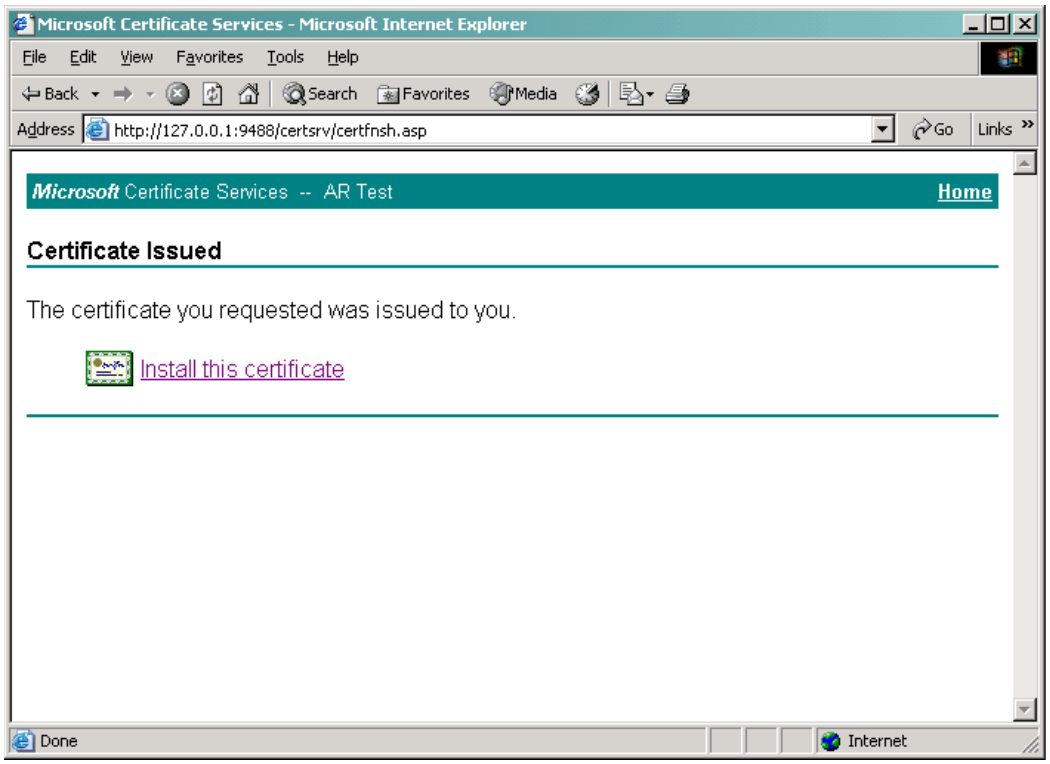
If the certificate you request has not yet been granted, the Certificate Pending window displays as shown in Figure 2-10. This window provides a button to remove the certificate request.

Figure 2-10 Certificate Pending with Remove Option



Assuming that the certificate request was approved, the server displays the Certificate Issued window shown in Figure 2-11.

Figure 2-11 Certificate Issued

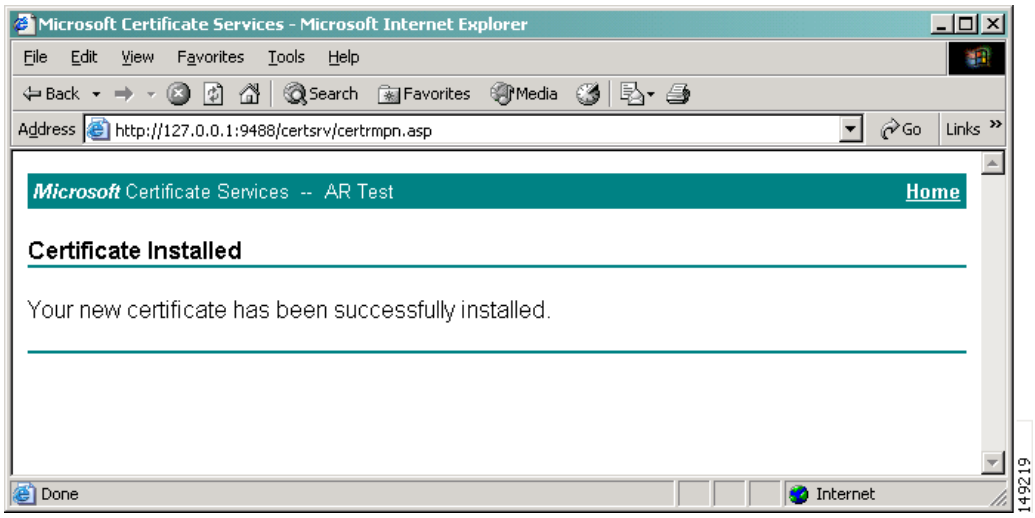


Step 9 Click **Install this certificate** to continue.

Note Depending on your configuration, you might be asked to confirm your request.

Figure 2-12 shows a confirmation of successful certificate installation.

Figure 2-12 Certificate Installed Confirmation



Exporting Server and Client Certificates

The certificate and private key must be exported from Windows before they can be installed on another machine. The easiest way to do this is to use the browser. This example uses Internet Explorer version 5.0. The procedure is the same for server and client certificates.

Navigate to the **Certificates** dialog box with these steps:

Step 1 Open the Tools menu.

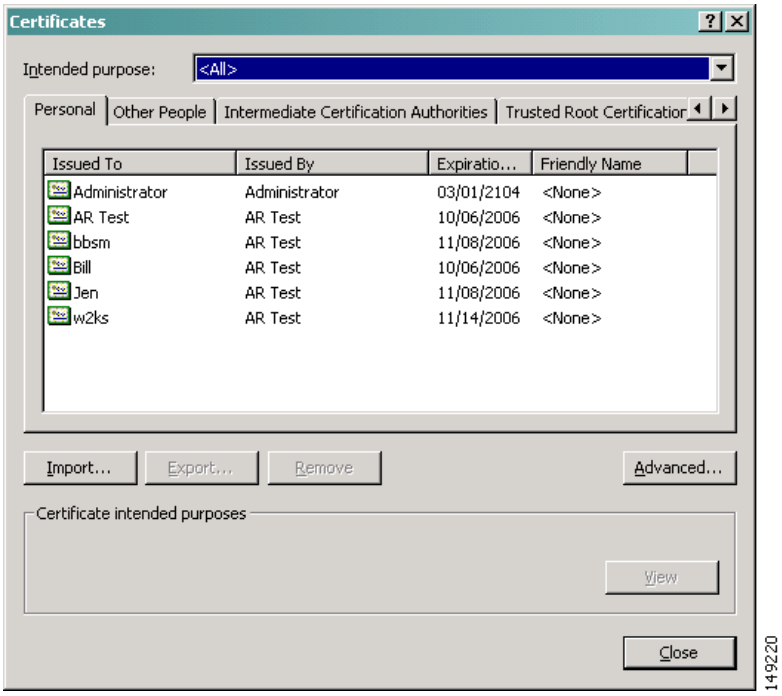
Step 2 Select **Internet Options…**

Step 3 Click the Content tab.

Step 4 Click **Certificates**.

The Certificates dialog box will display the certificates installed on this computer, as shown in Figure 2-13.

Figure 2-13 Certificates Dialog

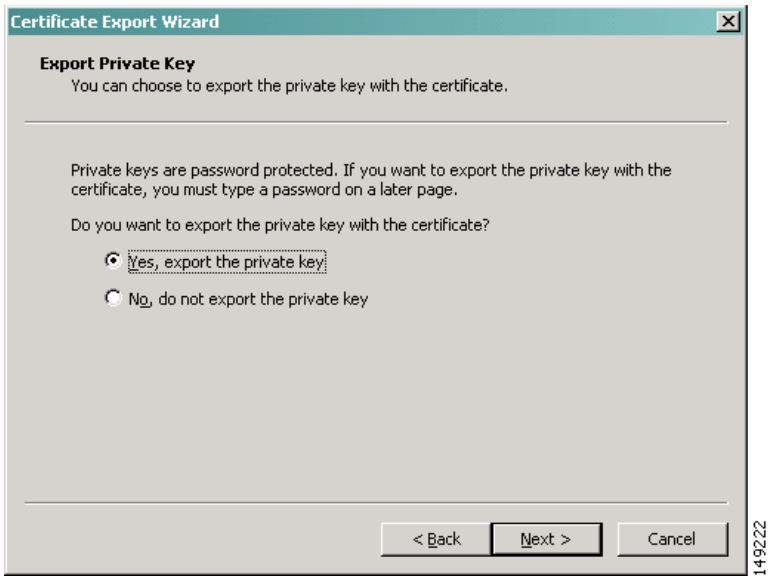


Step 5 Select the appropriate certificate to export and click **Export…** to initiate the Certificate Export Wizard as shown in Figure 2-14.

Figure 2-14 Certificate Export Wizard

Click **Next** to continue. The Export Private Key window displays as shown in Figure 2-15, which enables you to export the private key with the certificate.

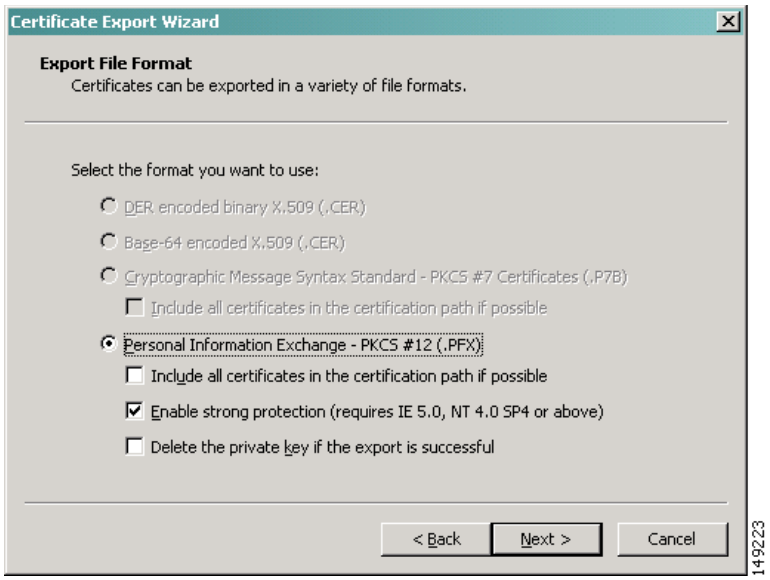
Figure 2-15 Export Private Key



Step 6 Select **Yes, export the private key** and click **Next** to continue.

The next window allows you to select the format of the certificate file. Since we are exporting both the certificate and the private key, the only format permitted is PKCS#12.

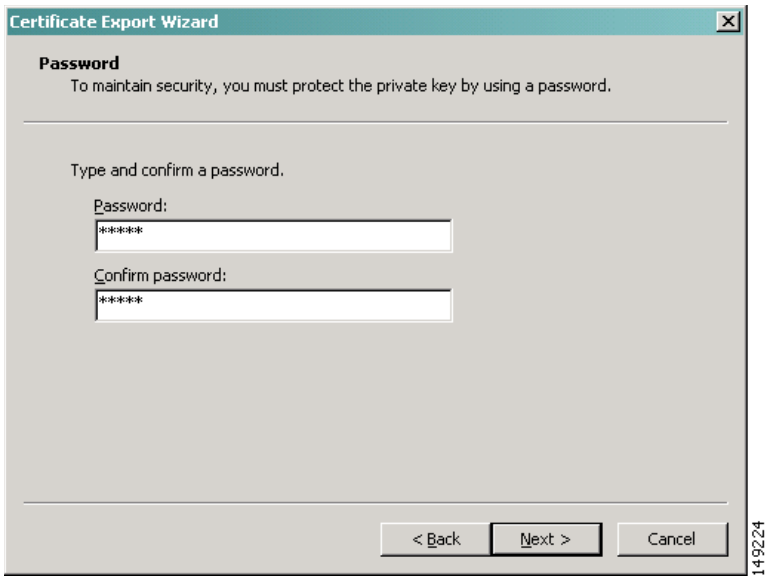
Figure 2-16 Export File Format



Step 7 Click **Next** to continue.

The next screen prompts you to enter the password used to protect the PKCS#12 content.

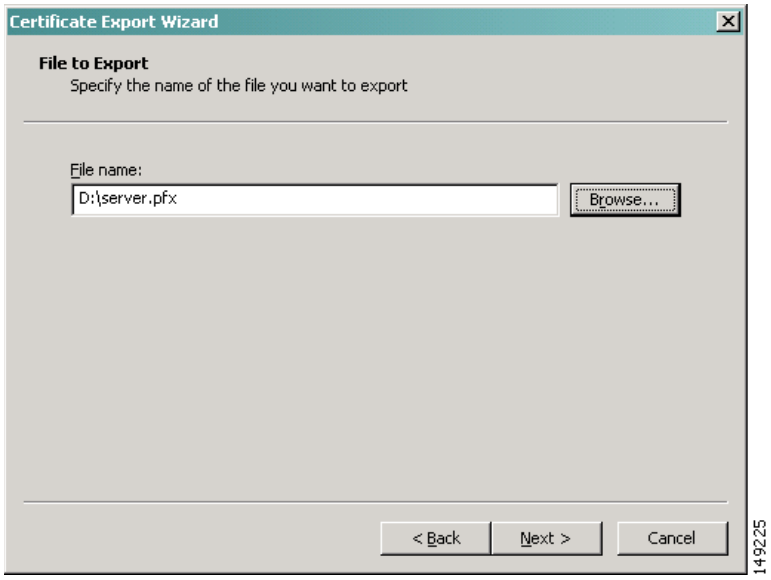
Figure 2-17 Export Wizard Password



After entering the password, click **Next** to continue.

The next screen prompts you to specify (or browse to) the name of file to export.

Figure 2-18 File to Export



Step 8 After entering the file name, click **Next** to continue.

Figure 2-19 shows the settings selected through the Certificate Export wizard.

Figure 2-19 Completing the Certificate Export

Step 9 Click **Finish** to complete the export operation.

If successful, the message shown in Figure 2-20 displays to indicate a successful export.

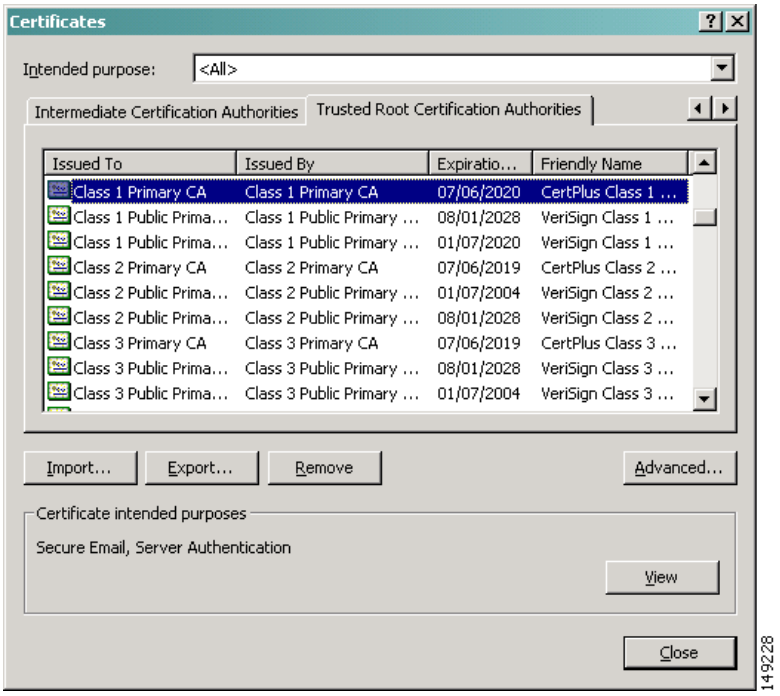
Figure 2-20 Successful Export

Exporting CA Certificates

The procedure shown in Exporting Server and Client Certificates, page 2-16, can also be used to export CA certificates. In the Certificates dialog box, click the appropriate tab for either Intermediate Certification Authorities or Trusted Root Certification Authorities and scroll the list to locate the certificate you want to export. (Other categories might also be present depending on how the certificate store is configured.)

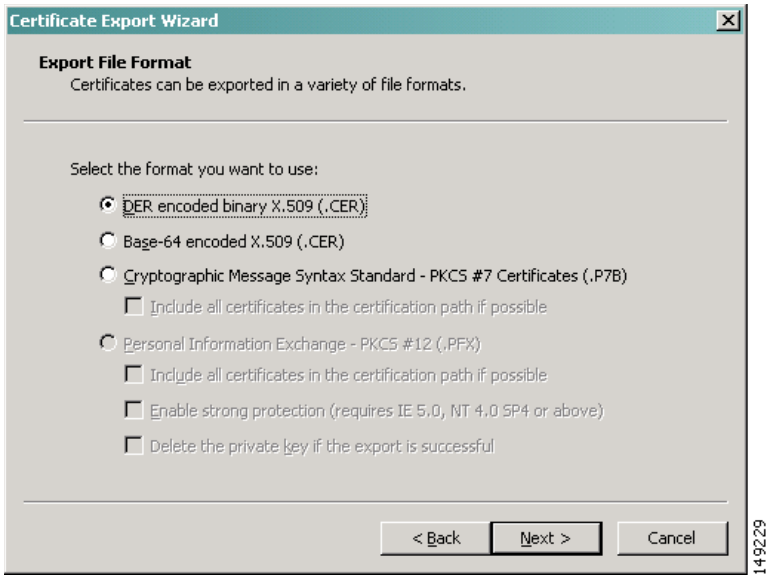
Figure 2-21 shows an example of the certificates to export window.

Figure 2-21 Certificates to Export



The rest of the procedure is the same as for server and client certificates except that usually the private key of CA certificates are not exported. In that case, the enabled exported certificate file format options will be a little different. Instead of just PKCS#12, the two formats will be DER or Base-64 encoded DER as shown in Figure 2-22.

Figure 2-22 Export File Formats



Converting PKCS#12 to PEM

A certificate and private key that have been exported from Windows will generally be encapsulated in a single file in PKCS#12 format. Before they can be installed on WLSE, they must be reformatted into PEM formatted files. It is a good security practice to store the certificate and private key in separate files. You can use the **openssl** command line tool from the OpenSSL open source project to do this.

Most Windows systems will not have the **openssl** tool installed, so it is usually easier to copy the PKCS#12 certificate file to a computer that does support OpenSSL.

There are two steps involved:

Step 1 Convert the PKCS#12 file to PEM format using **openssl**.

Step 2 Split the resulting PEM file into separate certificate and private key files (optional).

The following command line converts the content of **server.pfx** from PKCS#12 to PEM and places the result into **server.pem**:

openssl pkcs12 –in server.pfx –out server.pem

If the PKCS#12 file has been password protected, **openssl** will prompt you for the password. If the conversion succeeds, the **server.pem** file will contain the certificate and private key in PEM format. An example of a converted PEM file follows:

```
Bag Attributes
    localKeyID: 01 00 00 00
    1.3.6.1.4.1.311.17.1: Microsoft Base Cryptographic Provider v1.0
    friendlyName: 9191ccb399024e88287768944c8053cc_e0808bc1-0ea6-4702-85a9-1ccdee37a5c7
Key Attributes
    X509v3 Key Usage: 10
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,B22C8C19F46F9824

CcZ1Y5B5wu0l3bs/4wCv01RutRBLaRMdWb4QyRfPt6CpWnxZDkgvDVRyXnLiaw+g
75zYbWCMFMLsLiUeDr7yp0gSjpsOihwpdY4MEZL3V35iGIBl/dxOZ0dhywz8YoCo
D1XGQXBPTu3Uun/tZeyxISQDQFeGKopg1RvJzCdCEfF3hlg9bIWfFNuRhIOlvZoV
KeEO1TjsqtbH9hZTUyLH0dzfsix+xEw4Assc03KB4bEse+6Uk4Q3H7qFSwQNluxM
8zLpXMcIxSE0I+i9OdLbshUwBhKuW9/ksvvmneuaL7GCQRVvYj+QQmzAFPm0gYch
zDdBm+E86P8+J2/KlxEBWrFSpbgwpr6wM5SmqETH6VOIy2xaT+VTspDnp2jtJfyl
2/fOrG4V2njxuutQfGK9S2W49uVlcp1oss6iF4e7hVIY0+G1c/6LS8QFcQJoLbmw
ymmFRgi8j0jYGQWNpj7Awax+2Bpa+0wuWb/teCMX77/HRIJna+r8J0FbSWm3Z2JV
HAAvmtxO8SDJpuWSid1AVp7CFPESSpQHHy77dSFnGN3GBDmQTgUonbdN1ESCIMBM
LBIbaokDRt2HuoJXCaWa9vot2ssimidYlz0sOoTO2hHr8ai0dQErDGao5jAFzTGq
drKf+jA04GOagjknMfrNAfnnsOU1rKJAeW0oa24Z3YLep0LNhrf5H0ZuvrbDmv5x
KBlgCONuNNDcuEffjY6f/9MVpZX6dM80jjtD71xE/GR0oNUCsOhFTJZwGLSTJPS0
p3wxFswLNmqB2cWZgUifYX+UcQxkjb5quEeAqhY2UJujGu+yP12s9A==
-----END RSA PRIVATE KEY-----
Bag Attributes
    localKeyID: 01 00 00 00
subject=/C=US/ST=WA/L=Seattle/O=Engineering/OU=IT/CN=Server
issuer= /C=US/ST=WA/L=Seattle/O=Engineering/OU=IT/CN=Root CA
-----BEGIN CERTIFICATE-----
MIIDyzCCA3WgAwIBAgIKYbFeTwAAAAAAAzANBgkqhkiG9w0BAQUFADBhMQswCQYD
VQQGEwJVUzELMAkGA1UECBMCV0ExEDAOBgNVBAcTB1NlYXR0bGUxFDASBgNVBAoT
C0VuZ2luZWVyaW5nMQswCQYDVQQLEwJJVDEQMA4GA1UEAxMHUm9vdCBDQTAeFw0w
NTA5MjkxNzE4NTRaFw0wNjA5MjkxNzI4NTRaMGAxCzAJBgNVBAYTAlVTMQswCQYD
VQQIEwJXQTEQMA4GA1UEBxMHU2VhdHRsZTEUMBIGA1UEChMLRW5naW5lZXJpbmcx
CzAJBgNVBAsTAklUMQ8wDQYDVQQDEwZTZXJ2ZXIwgZ8wDQYJKoZIhvcNAQEBBQAD
gY0AMIGJAoGBALmJ4jD5IPDHS36RdmemhLCP0Q14bQzE8ngcFPcIsUY7YeK28tw2
GejsiAIE+kTEUdqPtVhqLT7NAQQykZjIzGNRpyiNuEJYDDsBow6J/SnHdPhTmAGJ
X0+YgCQdgb+3N6fSAOAv0APVcIY1/aN7cGhrSbYH4F11Nl0sfUSWLclZAgMBAAGj
ggHKMIIBxjAOBgNVHQ8BAf8EBAMCBPAwEwYDVR0lBAwwCgYIKwYBBQUHAwEwHQYD
VR0OBBYEFMo+Uu+Kxo8h/RWVtBLKTdIqxSC8MIGaBgNVHSMEgZIwgY+AFKD7ph9y
oLwfDhG6IWO+lIrZZd9ioWWkYzBhMQswCQYDVQQGEwJVUzELMAkGA1UECBMCV0Ex
EDAOBgNVBAcTB1NlYXR0bGUxFDASBgNVBAoTC0VuZ2luZWVyaW5nMQswCQYDVQQL
EwJJVDEQMA4GA1UEAxMHUm9vdCBDQYIQEkPt5Jo2DJ9J/drIo1njrjBjBgNVHR8E
XDBaMCqgKKAmhiRodHRwOi8vdzJrcy9DZXJ0RW5yb2xsL1Jvb3QlMjBDQS5jcmww
LKAqoCiGJmZpbGU6Ly9cXHcya3NcQ2VydEVucm9sbFxSb290JTIwQ0EuY3JsMH4G
CCsGAQUFBwEBBHIwcDA1BggrBgEFBQcwAoYpaHR0cDovL3cya3MvQ2VydEVucm9s
bC93MmtzX1Jvb3QlMjBDQS5jcnQwNwYIKwYBBQUHMAKGK2ZpbGU6Ly9cXHcya3Nc
Q2VydEVucm9sbFx3MmtzX1Jvb3QlMjBDQS5jcnQwDQYJKoZIhvcNAQEFBQADQQDT
mBDMuK1OZR7p8TtsEGH8G3qa7QnHtPGfEtaW5iFq72eqyQNx0lys7sl36iY/+S98
we3LO2Vyo9QegXLm5tv9
-----END CERTIFICATE-----
```

You might want to split the certificate and private key into two separate PEM files for security purposes. Since PEM is a text format, you can use any common editor (such as **vi** or **emacs**) to derive, for example, **server-cert.pem** and **server-key.pem** from **server.pem**.